HIJSON: a cartographic document format for web modeling of interactive indoor mapping

Marco Virgadamo
Dipartimento di Ingegneria
Università Roma Tre
Rome, Italy
virgadamo@dia.uniroma3.it

Marco Sportillo
Dipartimento di Ingegneria
Università Roma Tre
Rome, Italy
sportillo@dia.uniroma3.it

Federico Spini
Dipartimento di Ingegneria
Università Roma Tre
Rome, Italy
spini@dia.uniroma3.it

Alberto Paoluzzi
Dip. di Matematica e Fisica
Università Roma Tre
Rome, Italy
paoluzzi@dia.uniroma3.it

Enrico Marino
Dipartimento di Ingegneria
Università Roma Tre
Rome, Italy
marino@dia.uniroma3.it

Antonio Bottaro
Sogei S.p.A.
Ricerca e Sviluppo
Rome, Italy
abottaro@sogei.it

ABSTRACT

This paper introduces HIJSON[1], a novel indoor cartographic document format. A software framework is also presented, that relies on HIJSON documents and is entirely based on web technologies. With respect to current cartographic formats, HIJSON brings four major enhancements: (a) exposes a hierarchical structure; (b) uses local metric coordinate systems; (c) may import external geometric models; (d) accepts semantic extensions. The HIJSON format is designed to describe any geometry of the *indoor space* of complex buildings, capturing their hierarchical structure, a complete representation of their topology, and all the objects (either smart or not) contained inside. The textual representation allows the software framework to offer a web environment in which the user is presented with either 2D or 3D models of the indoor ambient to navigate. Such virtually rebuilt environment, accessible via web browsers from any kind of device, can be regarded as the platform where several applications may coexist: IoT monitoring; realtime multi-person tracking; cross-storey user navigation, through an algorithm that automatically finds valid walkable routes, taking into account both architectural obstacles and furniture. The semantic extensions supported by the HIJSON framework architecture encapsulate the details about communication protocols, rendering style, and exchanged and displayed information, allowing the HIJSON format to be extended with any sort of models of objects, sensors or behaviors.

[1]This work was partially funded with grants by SOGEI,the ICT company of the Italian Ministry of Economy and Finance.

DocEng 2015 Lausanne, Switzerland

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Algorithms, Design, Documentation, Standardization

Keywords

Document format, indoor cartography, indoor navigation, IoT monitoring, HIJSON, multiperson tracking, web application

1. INTRODUCTION

An indoor environment interactive mapping consists of a virtual reconstruction of a real indoor space in which the user can move around and then interact with the objects which are found the same position they actually occupy in the real world. Such an indoor interactive mapping can be thought as a specialized and very evoluted user interface capable of giving a glimpse of a section of the real world that the user can handle in a natural and intuitive way. Such a reconstructed virtual indoor environment can be considered a general platform where many different applications can rely upon. Both promising and already well explored ICT fields may find in *virtual indoor mapping* the perfect context to be integrated into.

In particular, for environments which massive presence of sensor-equipped (or "smart") objects, which realize the so-called *IoT* (Internet of Things), the indoor interactive mapping represents an ideal integrated interface for IoT monitoring systems. In particular, it can be the container of indoor navigation systems, giving the user, to be routed across an indoor environment, the opportunity to interact with objects along the suggested paths. Futhermore, in conjunction with the advancements in the fields of user indoor location, achieved via a variety of positioning systems like GNSS (Global Navigation Satellite system), Wi-Fi, LTE (Long Term Evolution), SDR(Software Defined Radio), it represents the most natural interface to perform realtime access monitoring and multiperson tracking.

To enable such a interactive mapping platform it is of the utmost importance a descriptive representation of the indoor environment. This means entering in the field of indoor cartography, which as digital evolution of plain floor plans, is arrived to arouse the interest of big players like Google, that has integrated into Google Maps indoor plans of specific locations of interest [10]. In general, can be considered "of interest" — such to justify and motivate indoor cartographic applications — both public or commercial places of vast dimensions, as for example airports, train stations, shopping malls, and also private buildings subject to strict access control, like warehouses, logistics centers, and datacenter.

Despite of the growing attention around indoor cartography, efforts to specify open formats for indoor representation are few and partial, certainly not intended to support the interactive indoor mapping, which is conversely the main purpose of this paper.

This work, jointly developed by Sogei S.p.A., a company fully owned by Italian Ministry of Economy and Finance, and the CVDLAB (Computational Visual Design Laboratory) of the "Roma Tre" University, is inspired by the necessities of SoGeI itself, which runs one of the largest data center of Italy, then requiring strict access control policies which have to be composed with indispensable human/machine coordinate maintenance scenarios. Support for these scenarios, where real time awarness of the maintainer position inside the data center helps to both reduce intervention and increase safety, has been adopted as the case study of the interactive indoor mapping based on the indoor cartographical format proposed.

The remainder of this document is organized as follows. In Section 1 we provide an overview of the state of the art in the field of indoor document standards and related applications. Section 3 is devoted to describe the advances introduced by the novel cartographic document proposed, while section 4 presents the document syntax. Section 5 reports about the toolkit specifically developed to handle the new document format. In Section 6 it is depicted the overall architecture and the implementation of the web based application framework, which is in turn used to achieve the objectives stated above. Section 7 presents a case-study application of the document format discussed in this paper. Finally, Section 8 proposes some conclusive remarks and future developments.

2. RELATED WORK

Researches about the cartographic representation of indoor environments are numerous and at the same time heterogeneos regarding to the strategies applied. Different information sources are used, and accuracy of the produced solution depends on the adopted approach. In some cases the information is obtained with automatic or semi-automatic processing of files that describe the architectural structure of a building, such as BIM (Building Information Modeling) [8] and/or IFC (Industry Foundation Classes) that describe a building project [4]. Image processing is also used to extract topological information from floor plan images [9]. In other works, building informations and descriptive parameters are redefined from scratch [11]. This choice is driven by the unsuitableness of the current representive formats: images contain poor information and CAD files are not designed for this kind of use.

A recurring theme among the use of carthograhic information is *indoor navigation* [9, 11, 4]. The proposed approaches are very different in this case too, and based on several strategies with some basic elements in common. An often adopted solution is based on the representation of the routing information as a graph, having a node for each room and an edge for each couple of connected rooms. In some cases the edges are weighted in function of euclidean distance. The detail level of the graph and hence the effective practicalbility of the calculated paths, can vary depending on the technique and the design choices applied, but in general most of the proposed solutions retrieve information only from architectual structure.

A subject related to navigation is the *location of users*. All the considered works agree on the inadequacy of GNSS in indoors contexts, due to the segnificant reduction in signal quality. To locate the exact position of a user inside a building, the currently most applied techniques are based on fingerprinting and triangulation of radio signals (WiFi, Bluetooth, etc.) flanked by more original solutions based, for example, on image recognition [1]. User tracking issue is faced with solutions that range from the clever utilization of inertial tracking sensors embedded in many smartphones [1] to the adoption of ad hoc devices [9].

The actual "de-facto" standard in terms of geospatial data representation is the *G*eoJSON format, which can be easily used for any type of geographical annotation. In some cases it has been slightly adapted to be used in indoor environments: it is the case of the *I*ndoorJSON format.

2.1 The GeoJSON format

GeoJSON is a geospatial data interchange format based on JSON, suitable for a geometrical encoding of various geographic data structures. As opposed to GIS format, GeoJSON is an open standard. Positions need to be expressed in geographical coordinates (usually WGS84).

GeoJSON supports some geometric primitives, including: `Point`, `LineString`, `Polygon`, `MultiPoint`, `MultiLineString`, and `MultiPolygon`. Lists of geometries are represented by a `GeometryCollection`. Geometries with additional properties are `Feature` objects. Lists of `Feature` are represented by a `FeatureCollection`.

A single `Feature` is composed essentially by two mandatory fields: `geometry`, which describes the objects' geometry accordingly to the previously defined primitives, and `properties` which contains additional information about the `Feature`.

It is possible to define complex shapes through the composition of simple GeoJSON objects. Mainly due to its semplicity, GeoJSON is widely used and deeply integrated in several applications and services.

2.2 The IndoorJSON format

IndoorJSON is a GeoJSON variant defined and used by *indoor.io*, a Finnish company devoted to indoor environment mapping. IndoorJSON is compliant with GeoJSON syntax, and it may consist of any number of `Features` and/or `FeatureCollections`. All `Features` are interpreted similarly regardless of their grouping into nested `FeatureCollections`. IndoorJSON supports all GeoJSON geometry types.

Some particular properties are used to correctly define indoor elements:

- `level`: described which level contains the feature;
- `geomType`: identifies the object's category, useful during the visualization process.

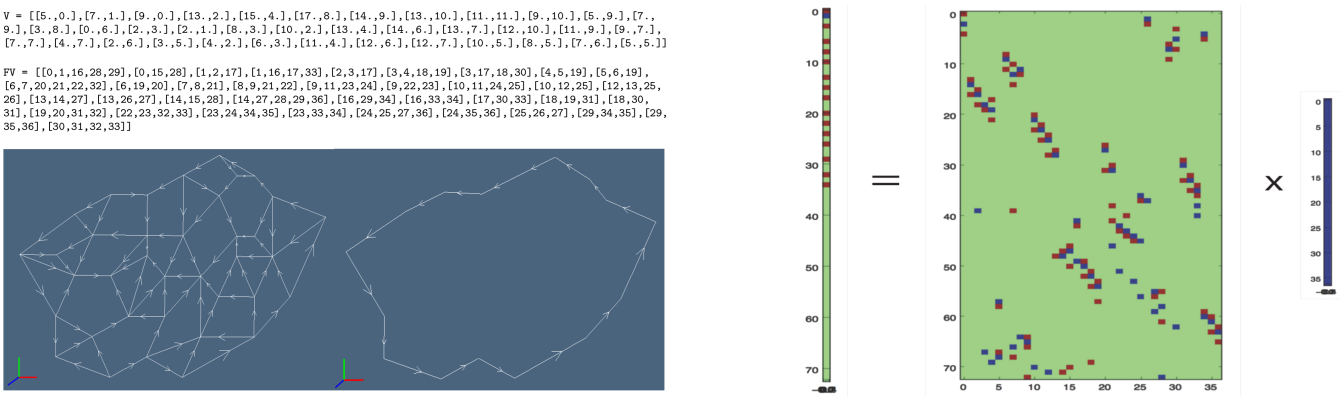


Figure 1: A toy example of the LAR scheme: (a) the bare minimum of data with *complete* information about topology; (b) the extracted boundary; (c) the extraction method $[e] = [\partial][f]$ giving the coordinate representation (in the discrete basis of the 1-cells) ofthe boundary edges $[e]$ by product of the sparse boundary operator matrix $[\partial]$ times the coordinate representation $[f]$ of the 2-cells (faces), in the discrete basis of the 2-cells.

There are some additional not mandatories properties useful for the indoor representation:

- `accessible`: describes if an element is walkable or not;
- `connector`: defines if the element is a connection between two levels;
- `direction`: describes the direction of the connection (both ways, only up, only down);

A syntax validator is provided by *indoor.io*, but the commercial nature of this project limits the number of tools available to deal with this format.

3. ADVANCES ON CARTOGRAPHIC DOCUMENT FORMATS

The focus of this work is the definition of a novel format of cartographic documents along with the software ecosystem rooted on it. A simple but effective algorithm to find indoor valid routes is also provided. The HIJSON (**H**ierarchical **I**ndoor **JSON**, this is the name chosen for the format) and the accompanying software framework aim to realize a mapping of indoor real spaces with a virtual interactive web environment. The HIJSON is based upon ideas and design principles collected from previous formats and identifies four critical improvements with respect to them: it exposes a *hierarchical structure*, uses *metric local coordinate system imports external hyperlinked geometric models* and accepts *semantic extensions*.

3.1 Hierarchical structure

The HIJSON format allows for hierarchical description of indoor spaces. The introduction of a hierarchical structure establishes a parent-child relation between entities of the model, reflecting a container-contained relationship. This direclty implies a neater representation than the plain linear structure adopted by GeoJSON, being a perfect analogy of objects contained (i.e. placed) into spaces.

In addition, more organized arrangement is allowed by logical (or even physical) grouping: concepts like building wings, sections, stories, departemens, etc. can be introduced to reflect into the document structure logical or physical real divisions, categories or relashionships.

Hierarchical structures are common in computer graphics since they are used as scenegraphs. This accordance of underlying structures really simplify 3D render algorithms of HIJSON documented environments.

Furthermore the container-contained relation enables to use local reference system.

3.2 Metric local coordinate system

Supported by the hierarchical underlying structure, the HJSON document format allows for the use of local coordinate system. This means that the shape of all elements can be convenient modelled using local coordinate and than placed in the right position with respect to the position of the parent (or container) element applying a translation and a rotation vector.

Another substantial advantage is represented by the adoption of a metric reference, consequently simplifying the compilation of the document, however it is, manual or aided by software tools.

The HIJSON document format is specially designed to guarantee the user to be routed seamlessly from outdoor to indoor and vice versa. Even though indoor geometries are inputted in a metric local continuous outdoor-indoor navigation is ensured via the definition of a processing pipeline detailed in the following.

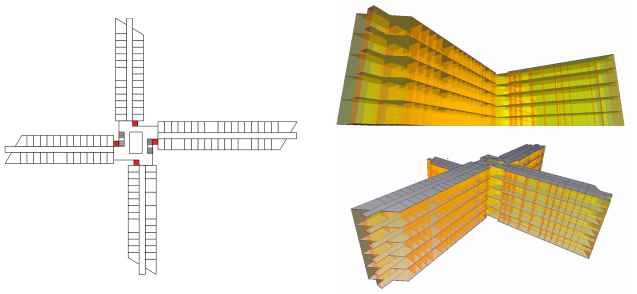
3.3 Hyperlinked geometric models

A HIJSON document may further import external geometric models—either of the buildings itself or the interior furniture or devices—that are topologically complete (in the sense of solid modeling [13]) and very compact. Such models coming from a source outside the document are acquired by linking JSON files that contain a Linear Algebraic Representation (LAR) of topology and geometry, to be expanded for visualization or interaction at any useful level of detail.

The LAR scheme [7] is characterised by a very large domain, including architecture, building and construction [12], 2D and 3D engineering meshes, non-manifold geometric and solid models and meshes, and high-resolution 3D images [6]. This scheme uses the set of *Combinatorial Cellular Complexes* (CCC) as mathematical domain [2], and various compressed representations of *sparse matrices* [5] as codomain.

Since LAR provides a complete representation of the topology of the represented space, the matrix $[\partial_d]$ of the boundary operator shall be used to compute the coordinate representation $[c]$ of the *boundary* chain of *any subset* c of cells, though *a single* operation of SpMV multiplication [5] between the `CSR` (Compressed Sparse Row) representation of $[\partial]$ and the `CSC` (Compressed Sparse Column) representation of the $[c]$ chain, resulting in very efficient computations on modern hardware, even mobile.

The expansion of a LAR model, to be considered as a general-purpose graphic primitive, may be executed on either the server or the supervisor client of the HIJSON Web Toolkit architecture (see Section 6.2.1), or even on the *Explorer* client, depending on the size and the locality of the model to be expanded.



(a) (b)

Figure 2: Office building: (a) the schematic plan; (b) the simplified 3D model generated for testing on the field the in-door mapping project described in this paper.

3.4 Semantic extensions

Semantic extensions make the HIJSON format extendible and customizable, that is, able to adequately respond to any need of objects representation. To define a semantic extension means to allow the HIJSON document to model an object previously not covered, or even to modify the behaviour of a comprised one. Semantic extensions are to be defined both as HIJSON format syntyax and as HIJSON Toolk source code. In particular it is necessary to define respectively a new HIJSON Element and a new HIJSON Class, as specified below.

4. HIJSON STRUCTURE AND SYNTAX

Listing 1 shows a simplified HIJSON document, devoid of puctual datails, to make clear to the reader the overall document structure.

```
{
  "config": {
    // ...
  },
  "data": [
    ...
    {
      "id": "architecture",
      "type": "FeatureCollection",
      "features": [
        // ...
      ]
    },
    {
      "id": "furniture_1",
      "type": "FeatureCollection",
      "features": [
        // ...
      ]
    },
    // ...
  ]
}
```

Listing 1: Example of HIJSON document.

The HIJSON document is composed of a configuration section, followed by one or more `DataCollections`, containing the actual data.

The configuration includes parameters and settings needed for building representation in the form of a JSON Object. One of the core information in this section is defined by the correspondence between three points of the local coordinate system and three point of the real world, expressed in geographical coordinates. This is needed to ensure a seamlessly passage from local to geographical coordinate system and vice versa.

After the configuration part, goes a list of `DataCollection`. An example of `DataCollection` is given in listing 2.

```
{
  "id": "architecture",
  "type": "FeatureCollection",
  "features": [
    // ...
    {
      "type": "Feature",
      "id": "room_0.1",
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [ [0, 0], [11, 0], [11, 19], [0, 19] ]
        ]
    },
    "properties": {
      "class": "room",
      "parent": "level_0",
      "description": "Office of Mr. Smith",
      "tVector": [10, 20, 0],
      "rVector": [0, 0, 90]
    },
    // ...
  ]
}
```

Listing 2: Example of `DataCollection`.

Each element of the list is given in the form of a GeoJSON `FeatureCollection`, containing an arbitrary number of *HIJSON Elements*. Each `DataCollection` imposes a logical relationship that can be exploited to group together related HIJSON Elements. Since HIJSON Elements adhere to the GeoJSON format, each `DataCollection` results compliant with GeoJSON syntax and then accepted by any GeoJSON validator. As detailed below, the HIJSON format introduces some additional rules that allow the adoption of this format for indoor representation.

4.1 HIJSON Element

Dealing with indoor environments, there are essentially two classes of object that is necessary to represent. They are (a) architectural elements, like a room, a corridor, a wall, etc. and (b) furnishings, intended in a broad sense, such as to contain so furniture, like a desk or a chair, as “smart objects” like an IP-cam or a thermostat.

An HIJSON Element defines a syntax to describe both geometry and properties of an object and represents the atomic component of an HIJSON document. It is in turn compliant with GeoJSON syntax. It would be a best practice to group together related JSON Element using `Data Collection`: several strategies can be applied, for example grouping by storey or even by room can be imposed. Alternatively, since the furnishings are more likely to change than the architectural components of a building, these two different kind of elements can be isolated in different `Data Collections`.

The hirarchical structure of the document is embodied by the possibility of the HIJSON Elements to have childern elements, so a unique ID is mandatory for each HIJSON Element.

There are three allowed Geometry types that can be used: `LineString`, `Point`, and `Polygon`. The choice of a Geometry type to associate to a HIJSON Element implicitly define the category of the element: `LineString` are used for walls and doors, `Point` for funishings, and `Polygon` describes levels and rooms.

The Geometry coordinates are expressed in meters, and for convention starting at the bottom-left of the element. Unlike GeoJSON, where all the properties are optional, in HIJSON some striclty requirements are imposed and some attributes are mandatories:

- `class`: represent the element category, used to instantiate the appropriate *H*IJSON Class;
- `parent`: contains parent's ID of the element;
- `tVector`: represents the translation relative to the parent element, expres in meters;
- `rVector`: represents the rotarion relative to the parent element, expressed in degrees.

Specific classes may require the mandatory presence of other properties. For eexample the two classes `internal_wall` and `external_wall` that define internal and external walls respectively, require a `connections` array, containing the IDs of the adiacent elements. This information is used by the connector children of the element (e.g. like doors) to identify the areas linked together.

Given the nature of the GeoJSON format from which HIJSON derives, the elements are represented by their 2D shape, like on a planimetry. To assign a value to the height of the object, intended as third dimension, has been introduced the property `height`.

A `description` property can provide further information about the element.

Arbitrary optional fields can be added without restrictions, in order to enrich and extend the expressvity of the representation, or with the simple documenting purpose.

5. HIJSON TOOLKIT

The HIJSON Toolkit is a software module that implements common operations and transformations on HIJSON documents. Written in *JavaScript* language, it has been built to be deployed in the web environment. It is *modular* and entirely *isomorphic*, i.e. can run on the server as well as on every client. Working in the web environment, the Toolkit benefits of the fertility as regards the software development in this field: it takes advantage of libraries and frameworks such as *Ract*, "the JavaScript library for building user interfaces" by Facebook, and *Three.js* a framework to deal with *WebGL* tecnologies.

The Toolkit realizes the instantiation and extension logic of a HIJSON document, and realize a multistage transformation pipeline that, as required, can be used entirely or only in part.

5.1 Processing pipeline

The HIJSON processing pipeline relizes the sequence of preliminary transformations that have to be applied to a HIJSON document before any futher operation. It is not strictly required to complete each stage of the pipeline: the exit stage dipends on the specific use case.

The application of the transformation pipeline has a double aim. The first one consists in generating the graph of valid paths between all the interesting HIJSON elements. The second objective is the generation of one *GeoJSON* document for each storey of the building described by the HIJSON document. In this way a bidimensional plant for each level of the building can be provided and visualized through any compliant GeoJSON viewer.

HIJSON processing pipeline (as pictured in figure AGGIUNGERE RIFERIMENTO) is composed by 6 elaboration stages. In the following are detailed operations excecuted by each stage, which are, in the order: *validation*, *georeferencing*, *parsing*, *graph paths generation*, *2D layers generation*, *marshalling*.

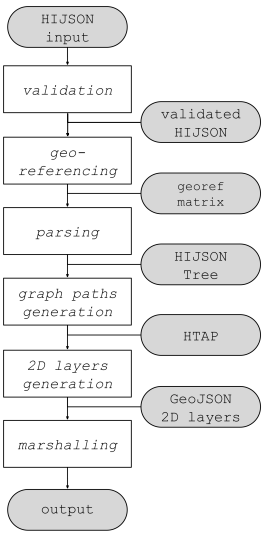


Figure 3: HIJSON processing pipeline

1. ***validation*** - The first one is the validation stage. In order to begin with the effective transformations the input HIJSON document must be compliant with the format syntax and structure requirements. In the case the validation stage fails, processing aborts and do not continue to following stages. If the stage success, the output for the next stage is a validated HIJSON.
2. ***georeferencing*** - In the second stage, in order to allow for continuous outdoor/indoor navigation, the system needs to compute the georeferencing matrix, a linear operator able to transform local coordinates into

global coordinates (referred to world coordinate system as latitude and longitude misures) and viceversa. This task is accomplished by solving a linear system obtained from information contained in HIJSON configuration part and precisely from the corrensponsdance of three real word points to three points included into the HIJSON document.

3. *parsing* - The parsing stage, takes the validated and georeferenced HIJSON as its input, that as illustrated before can be thought of as a list of HIJSON Elments, parses them and produce an istance of HIJSON Tree. The HIJSON Tree is an object in memory representing the tree hierarchical structure of the building described by the HIJSON document.
4. *graph paths generation* - The fourth stage is in charge of the generation of the graph paths. The algorithm to achieve such a goal is introduced below. The graph paths will be useful afterwards to compute valid paths from couple of point of interest on the graph. Once the graph paths has been computed, the input HIJSON Tree is augmented with paths information, becoming what has been called an HTAP (HIJSON Tree Augmented with Paths). Augmentation always takes place as leaf nodes added as children of a specific level (e.g. "room").
5. *2D layers generation* - The fifth stage is the generation of GeoJSON layers. For each storey of the building, the Toolkit generates one geoJSON layer that will be use for the creation of a 2D map. Each layer contains only the children of a 'level' node of the HIJSON Tree. The presence of a specific element inside the layer can be finely tuned by means of a boolean value. Every element has geographical coordinates calculated by the transformation matrix with regard to the local coordinates of the HIJSON Element.
6. *marshalling* - The last stage is responsible of execute a serialization of the the transformed data. Tasks like breaking dependency-loops and stringification are performed. This stage is useful mainly serverside, as and the output is stored ready to be served to any requiring client.

5.1.1 Algorithmics: automatic generation of valid paths

The fourth stage of the processing pipeline is responsible for the generation of a graph of valid paths through the entire model represented by the intput HIJSON document. The graph generated according to the algorithm described in the following, although not optimal, ensures a complete coverage of the surface while limiting the numebr of generated nodes. Resulting graph is weigthed on the edges with nodes distances and each node represents alternatively:

a. standard path node, i.e. a junction node or possibly an endpoint of a path;
b. connection node, used as subproblem composing element in the divide et impera approch adopted (as described below).
c. element nodes ie. HIJSON Element (whose HIJSON Class explicitly grants his presence in the graph), typically an endpoint of a path;

Such a graph allows for calculations of directions between any two given nodes. Although different approaches have been explored [3], a very classical solution has been selected in this case, so directions are actually computed clientside applying the Dijkstra's shortes route algorithm on the graph.

Taking advantage of the hierarchical structure of the HIJSON document, and according to the divide et impera approach, the problem of the graph paths generation is splitted in several sub-problems which consist in the computation of the sub-graphs relative to each room, or more generally ambience. The sub-graphs are then linked together through the connection nodes (which in most cases represents doors). The resolution of each sub-problem (as depicted in figure 4), is composed by 4 phases:

1. Computation of the walkable area of the ambience: this task is accomplished subtracting area of the possibly encumbrances to the area of the ambience; the result is tipically a surface with holes;
2. Triangulation of the walkable area: the computed surface is triangulated taking into account the presence of holes;
3. Identification of graph nodes: for each triangle side completely internal to the area, its midpoint is selected as standard path node;
4. Junction of nodes: nodes relative to the same triangle are then linked together; both element nodes and connection nodes (i.e. doors) are linked to the nearest node in the ambience (i.e. room).

5.2 HIJSON Class definition

To exploit the possibilities offered by HIJSON Toolkit, along with the HIJSON document, some custom dynamic behaviours must be described. These behaviours encapsulate the specificities relative to comminucation procols with the sensors as well as user interaction peculirities. The interface for these behaviours is the HIJSON Class.

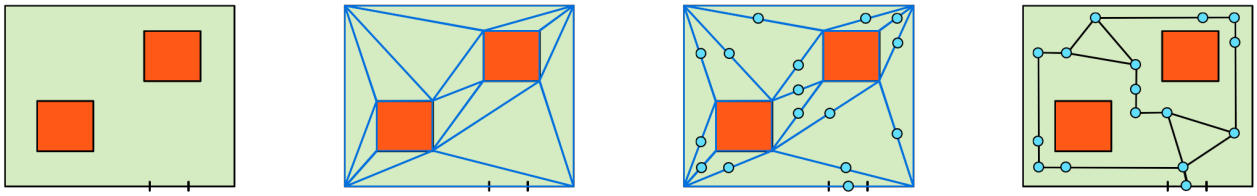
Figure 5: HIJSON Element/Class/Node relashionship

Each HIJSON Element of the HJSON document given as input, has a dynamic counterpart, a running instance called HIJSON Node, instantiated according to the corresponding HIJSON Class via relfection methods (see figure 5).

To specify a new HIJSON Class means to extend the Toolkit to deal with a new class of HIJSON Element.

To extend the toolkit to deal with a new class of HIJSON Element is required to to specify a new HIJSON Class, defining the following properties and methods:

- `in_graph`: a boolean value to express if the element is an approachable point in the graph paths;
- `in_2D_map`: a boolean value to express if the element has to be showd in the 2D map;
- `get2DStyle`: a method that returns the 2D map appearence of the element, essentially HTML and CSS code;
- `get3DModel`: a method that returns the 3D model appearence of the element, an instance of `THREE.Object3D` of *THREE.js* framework;



(a) (b) (c) (d)

Figure 4: Graph paths generation: (a) detection of obstacles and computation of walkable area; (b) triangulation of walkable area; (c) identification of graph nodes area; (d) junction of nodes.

- `getWidget`: a method that returns the information widget, a *React* component;
- `getProxy`: a method that returns server side proxy which encapsulate IoT sensor communication protocol, a *Node.js module*.

User's needs for new indoor elements, different sensor equipment, alternative representation on 2D or 3D viewport are accepted by the definition of new HIJSON Classes that allows in this way single point custom extension of the Toolkit capabilities.

6. HJSON WEB FRAMEWORK

The HIJSON Web Framework responds to the needs of an extendable, customizable, and scalable web framework which provides at the same time IoT monitoring, realtime multi-person tracking and crossfloor user navigation.

Expandability and customizability derives from both design choises and HIJSON inherent characteristics, the possibility of semantic extensions. Scalablility is directly borrowed from technologies used for the software development: *JavaScript* language, using *Node.js*, in particular *Express.js* as backend framework, exploiting the power of WebSocket protocol through the *Socket.io* library.

Being supported by the web as bearing platform, the framework exposes also an highly availability: it is so simple to use as to visit a website, both from desktop or mobile devices, without explicit requirements to install any software from proprietary stores (access to which is often denied from business devices).

The HIJSON Web Framework deeply relies on HIJSON Toolkit and offers the overal client/server architecture and a convinient, highly intractive user interface, leaving aside the specific indoor positioning system and the IoT sensors, to deal with a robust interface is provided and described in the following seciton.

6.1 Applications

The Framework has been designed focus on two different possible kind of users: the *Explorer* and the *Supervisor*. They have different requirements and are likely equipped with different devices: while the *Supervisor* monitors the indoor environment through a desktop workstation, the *Explorer* has a smartphone available and needs to be routed across the building.

In both cases, the web platform ensures a perfect alignment with the BYOD (Bring Your Own Device) approach, nowadays often supported by companies that encourage employees to use personal devices.

6.1.1 IoT monitoring

Every element in the HIJSON environment is capable of showing information about itself, so it can be analyzed by the user. The modularity of the HIJSON Toolkit permits to show particular information or UI about a specific object, using polymorphic behaviours of the different HIJSON Nodes. If an object is connected to the network and it is capable of interaction, the user can benefit of its functions through the system (e.g. if the object is a thermostat, the user can see the temperature in the room and can turn on/off the heating). If the object isn't interactive, the system can show static information (e.g. for fire Extinguisher, the system shows the last date of checking).

6.1.2 Realtime multi-person tracking

A typical task performed by a *Supervisor* can be the monitoring of users locations inside the building. This operation can be required for various reasons, e.g. security or logistics. The devices that equips the *Explorers* can be used to track their position in realtime, giving the *Supervisors* the whole picture of the presences inside the building in every moment.

6.1.3 Cross-storey user navigation

As shown in the algorithmics section, the HIJSON Toolkit provides a particular strategy to assemlbe a graph of possible paths inside the building. This graph, represented also in the form of a weighted adjacency matrix, can be easely used to compute paths between two nodes inside the building. To achieve this result, the matrix, which edges are weighted according to the distance between two nodes, is used as input in an applicaiton of the Dijkstra's algorithm. The result is the shortest path between two selected nodes of the graph. Thanks to the cross-storey connections of nodes representing stairs or elevators, the paths calculated can also start and end on different stories.

6.2 Architecture

Like the vast majority of the web based application, the Framework exposes an overall architecture that is inherently *client/server*. In particular, two different type of possible client are identifiable, one for each different kind of users: the *Supervisor* client and the *Explorer* client. Both of them connect to the same server.

The indoor space described by the HIJSON document passed as input is processed by the server via the processing pipeline. After that any connecting *Explorer* client, presum-

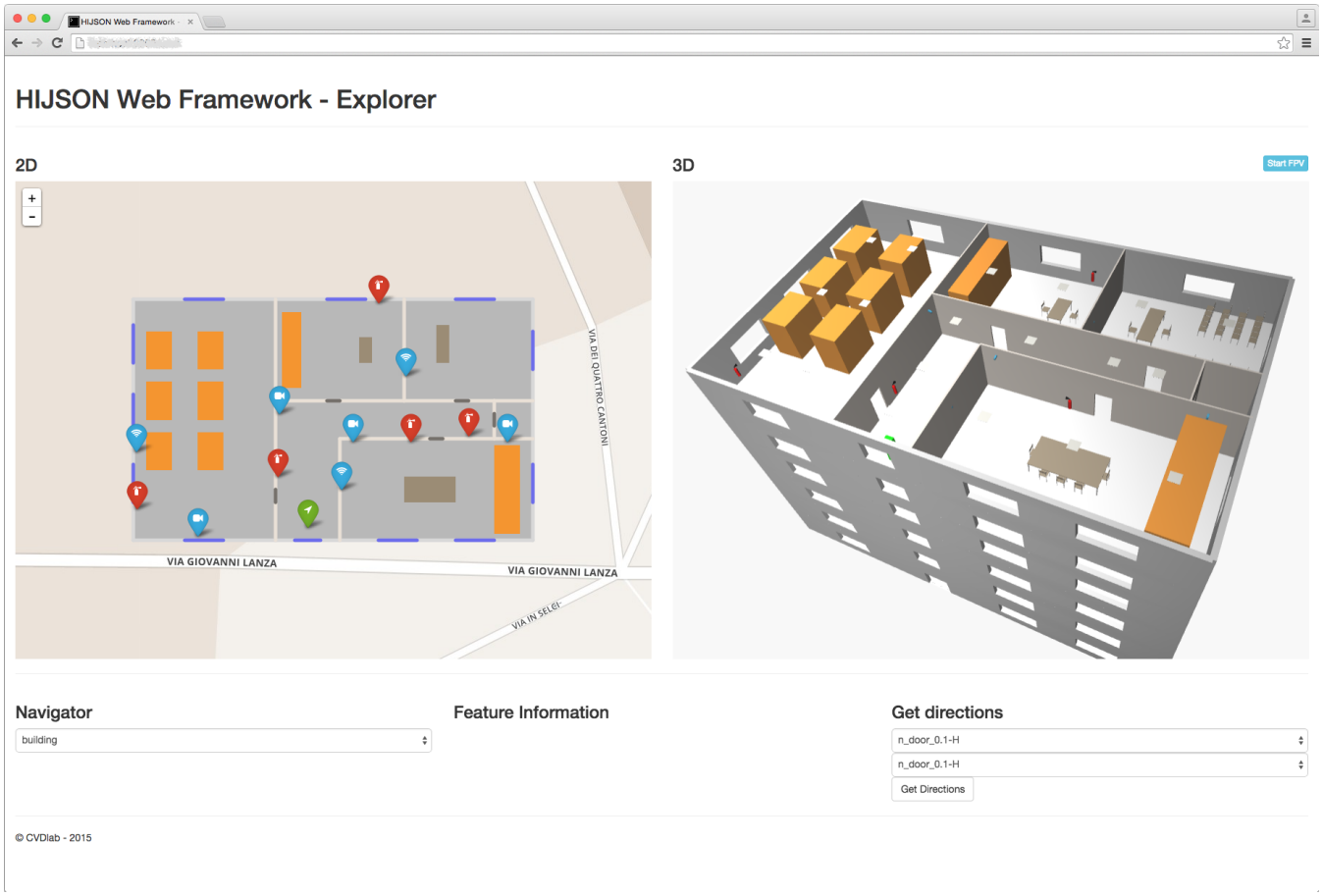


Figure 6: HIJSON Web Framework UI

ably through a mobile device, will be provided with the information to perform cross-storey navigation of the building, while reporting user position to the server. The server will feed any connecting *Supervisor* client with users positions, along with data from sensor- equipped objects present in the environment, realizing the IoT monitoring and the realtime multi-person tracking.

6.2.1 *Server Architecture*

The complete architectural scheme of the framework is provided in figure 7. A web server module is responsible for listenning to connecting clients. Each client connection is handled by the web server module providing all the required resources and opening one websocket channel, through which will flow *Explorer* and/or *Supervisor* communication protocol data. In particular, `multi-person tracking` module receives position data from *Explorer* clients. It aggregates and sends these information to connected *Supervisor* clients through the websocket channel, using a simple but realible protocol described later. Indipendence from particular IoT sensor equipment communication protocol is achieved introducing one `smart object proxy` module (defined in the HIJSON Class and obtained with `getProxy` method previously described) for each smart object modelled.

6.2.2 *Explorer client architecture*

The *Explorer* client architecture is generally deployed on a mobile device, which is usually supplied to a user who needs to be routed across the environment described by HIJSON document. The `sensor adapter` module encapsulates the communication logic with the indoor positioning system. The presence of this module ensures indipendece from particularly tecnology allowing client *Explorer* to rely on different indoor positioning systems (WiFi, LTE, etc.).

Every time the `sensor adapter` observe a perceptible modification in user position sends the new position information to the server through the single opened websocket, using the a message with the following syntax:

```
currentPosition = {
 coordinates: [x, y],
 levelId: level-ID
}
```

Relevant information includes, beside current coordinates, the indication of the storey of the possibly multilevel building the user is in.

It is to remark that, when not outflanked by the introduction of an external server, the problem of communication between positioning system and the low level APIs of the browser is left to positioning system itself or to whom is in charge of specific deployments of the HIJSON Web Framework.

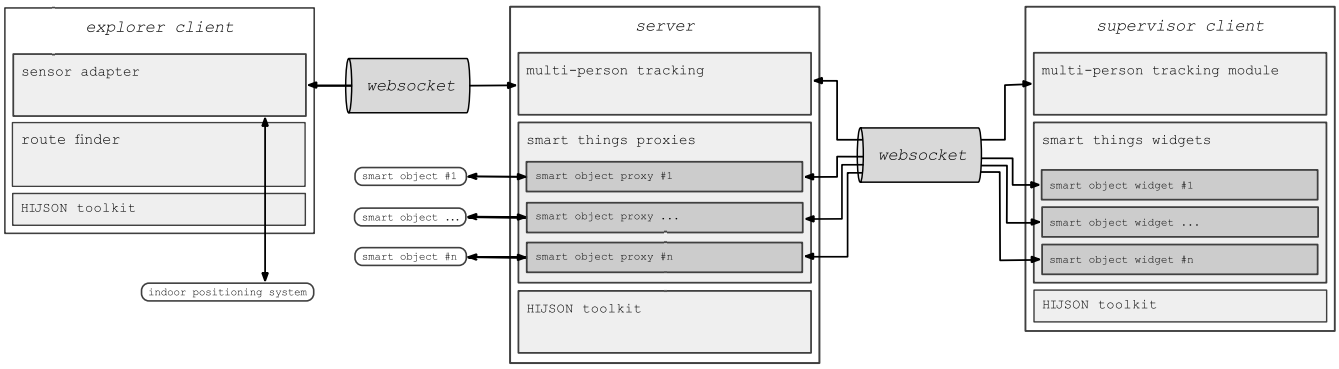


Figure 7: HIJSON Web Toolkit architecture

The `smart object widget` module, being in common with the client *Supervisor*, has been treated in the next section.

6.2.3 Supervisor client architecture

The Client *Supervisor* architecture shows two modules. The first one, the `multi-person tracking` module, is responsible to receive through the websocket, from the server information about explorers of the environment, showing them in the user interface. The second module, the `smart object widget` communicates with the server to propose to the user realtime information about sensor-equipped objects in the environment. Data passes through the single websocket opened between the server and every *Supervisor* Client. Rely on a naive but effective communication protocol, each `smart object widget` exchange data only with respective `smart pbject proxy` on the server. To ensure the the data is sent only when the user requires the information relative to a specific smart object, a widget lifecycle protocol is implemented: it is based on the 4 events `on_before_show`, `on_show`, `on_before_hide`, `on_hide` triggered, as suggested by their names when a widget is shown or hidden. When the user requires information about a smart object, its widget has to be rendered, but `on_before_show` the server is notified to connect via relative proxy to the sensor. Once connected, the server begin to send data via webscocket. Received data is shown through the wodget to the user. When done, that is the `on_before_hide` event of the widget is triggered, a notification is sent to the server announcing to stop sending data and the proxy close the connection to the sensor. Widget lifecycle protocol ensures that only requiring data is sent from th eserver to the client.

7. CASE STUDY

As the first case study has been taken into account the need of Sogei S.p.A. to support its data center maintaince, which is subject to strict access control policies. Lo scenario considerato consiste nell'intervanto di manutenzione da parte di un operatore che deve muoversi nell'ambiente e localizzare all'interno di un datacenter dalle enormi dimensioni la macchina su cui operare, individuandolo tra migliaia di rack di aspetto simile. Tale operatore sarà quindi equipaggiato con un client explorer. Il client explorer guiderà l'operatore fino al sottosistema su cui intervenire, notificandone al contempo la posizione ad ogni istante.

Le figure responsabili, attraverso il supervisor client possono monitorare la situazione degli smart object, che in questo caso spaziano dalle webcam ai sistemi di allarme antincendio ma comprendono anche le macchine fische, sicchè sia possibile monitorarne lo stato di funzionamento, temperatura di esercizio e carico di lavoro...

The realtime awareness of the relative positions between the 'maintenance man' andthe rack containing the machine will help us increasing safety and reducing intervention times. Infatti, le figure responsabili, con diritti di accesso al client supervisor, avranno quindi modo di monitorare le posizione degli operatori attivi all'interno del datacenter verificando che essi non devino su percorsi non autorizzati.

In particolare tali supervisori, nello scenario di manutenzione in analisi, tracciando l'intervento dell'operatore in tempo reale possono scaricare il sistema target della manutenzione da tutti i servizi che offre, migrando le corrsipondenti macchine virtuali e distribuendole su altri sistemi, garantendo così continuità di servizio e diminuendo il global risk factor. Terminato l'intervento tecnico, lo stato precedente può essere ripristinato.

8. CONCLUSIONS

In this paper a novel format for indoor cartographical description has been introduced, named HIJSON. Utilization of local metric coordinate system, avoiding the manipulation of geographical coordinate really inconvenient when dealing with indoor spaces and objects, greatly simplify the drawing up process of the document. Process that can be further improved realizing a graphical editor to assist the user during the description of the indoor space. The implementation of such an editor is already programmed.

The HIJSON format focuses on a hierarchical representation of the indoor spaces that allows for completely capturing their topology. On the basis of this representation a virtual web environment can be rebuilt working as a unifying platform to run a bunch of different applications. The reference architecture of such a platform has been also implemented and described in this work. The architecture supports a range of applications: IoT monitoring, realtime-multiperson tracking and user cross-storey navigation are already implemented and described. A very convenient way to extend representation capabilities of smart objects is also mentioned as semantic extensions. These extensions (which affects both document format and web framework) might be easly collected in a public repository. Community could both use public available extensions or contribute by mapping new (smart) objects inside the HIJSON document.

9. REFERENCES

[1] B. Al Delail, L. Weruaga, M. Zemerly, and J. Ng. Indoor localization and navigation using smartphones augmented reality and inertial tracking. In *Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on*, pages 929–932, Dec 2013.

[2] T. Basak. Combinatorial cell complexes and Poincaré duality. *Geometriae Dedicata*, 147(1):357–387, 2010.

[3] W. Bian, Y. Guo, and Q. Qiu. Research on personalized indoor routing algorithm. In *Distributed Computing and Applications to Business, Engineering and Science (DCABES), 2014 13th International Symposium on*, pages 275–277, Nov 2014.

[4] M. Boysen, C. De Haas, H. Lu, X. Xie, and A. Pilvinyte. Constructing indoor navigation systems from digital building information. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 1194–1197, March 2014.

[5] A. Buluç and J. R. Gilbert. Parallel sparse matrix-matrix multiplication and indexing: Implementation and experiments. *SIAM Journal of Scientific Computing (SISC)*, 34(4):170 – 191, 2012.

[6] A. DiCarlo, M. Jirik, and A. Paoluzzi. Cad models from medical images using the linear algebraic representation. In *CAD'15*, London, UK, June 22-25 2015. Accepted for publication in Computer-Aided Design and Applications, Taylor & Francis.

[7] A. Dicarlo, A. Paoluzzi, and V. Shapiro. Linear algebraic representation for topological structures. *Comput. Aided Des.*, 46:269–274, Jan. 2014.

[8] C. Eastman, P. Teicholz, R. Sacks, and K. Liston. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. Wiley Publishing, 2008.

[9] L. Faramondi, F. Inderst, S. Panzieri, and F. Pascucci. Hybrid map building for personal indoor navigation systems. In *Advanced Intelligent Mechatronics (AIM), 2014 IEEE/ASME International Conference on*, pages 646–651, July 2014.

[10] Google, Inc. Go inside with *Indoor Maps*. In `https://www.google.com/maps/about/partners/indoormaps`, 2014.

[11] D. Gotlib, M. Gnat, and J. Marciniak. The research on cartographical indoor presentation and indoor route modeling for navigation applications. In *Advanced Intelligent Mechatronics (AIM), 2014 IEEE/ASME International Conference on*, pages 1–7, Nov 2012.

[12] A. Paoluzzi, E. Marino, and F. Spini. Lar-abc, a representation of architectural geometry: From concept of spaces, to design of building fabric, to construction simulation. In Ph.Block, J.Knippers, W.Wang, and N.Mitra, editors, *Advances in Architectural Geometry*, LNCS (Lecture Notes in Computer Science). Springer, 2014.

[13] A. G. Requicha. Representations for rigid solids: Theory, methods, and systems. *ACM Comput. Surv.*, 12(4):437–464, Dec. 1980.